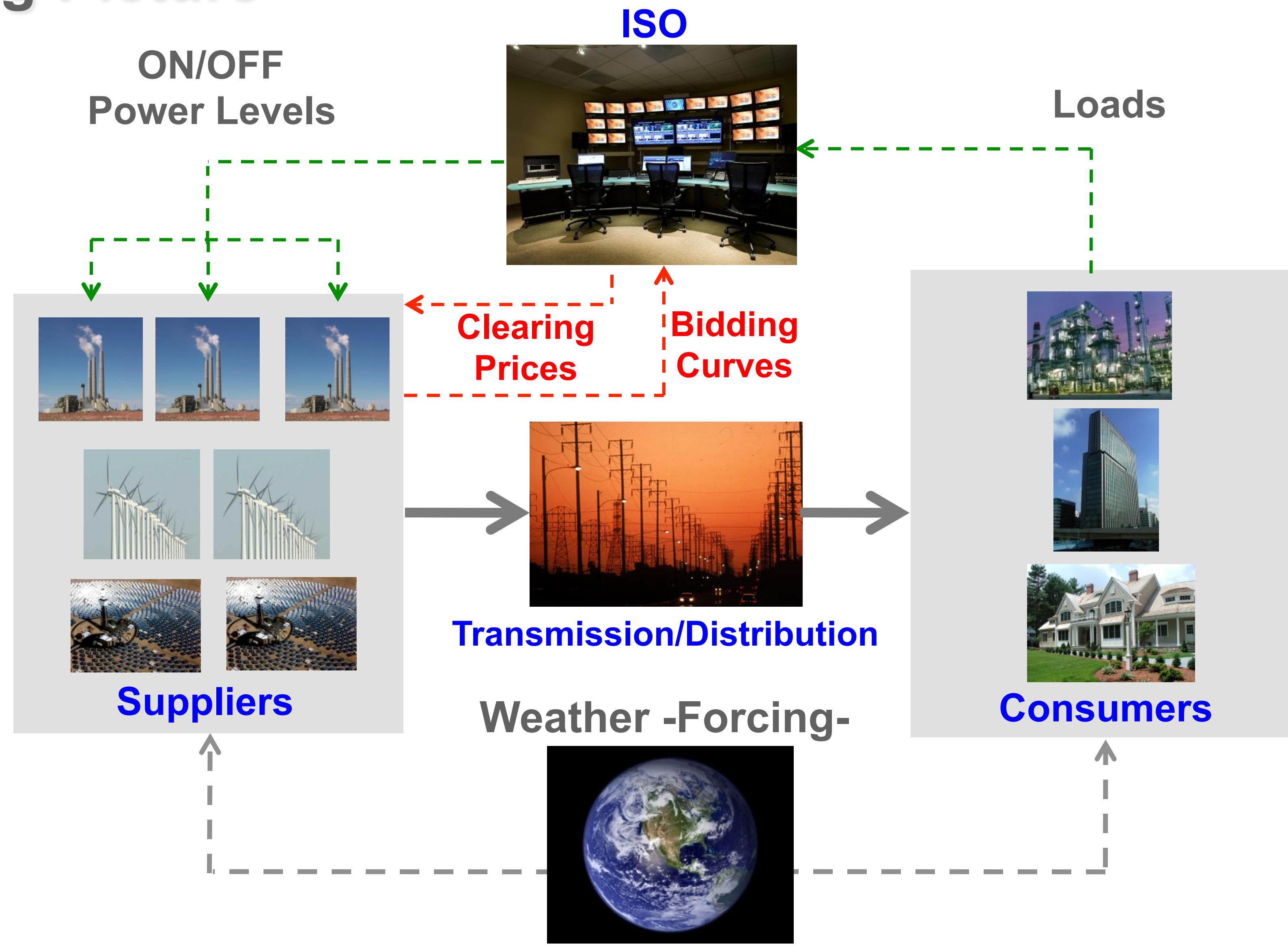
Scalable Dynamic Optimization

Mihai Anitescu
Department of Statistics
University of Chicago

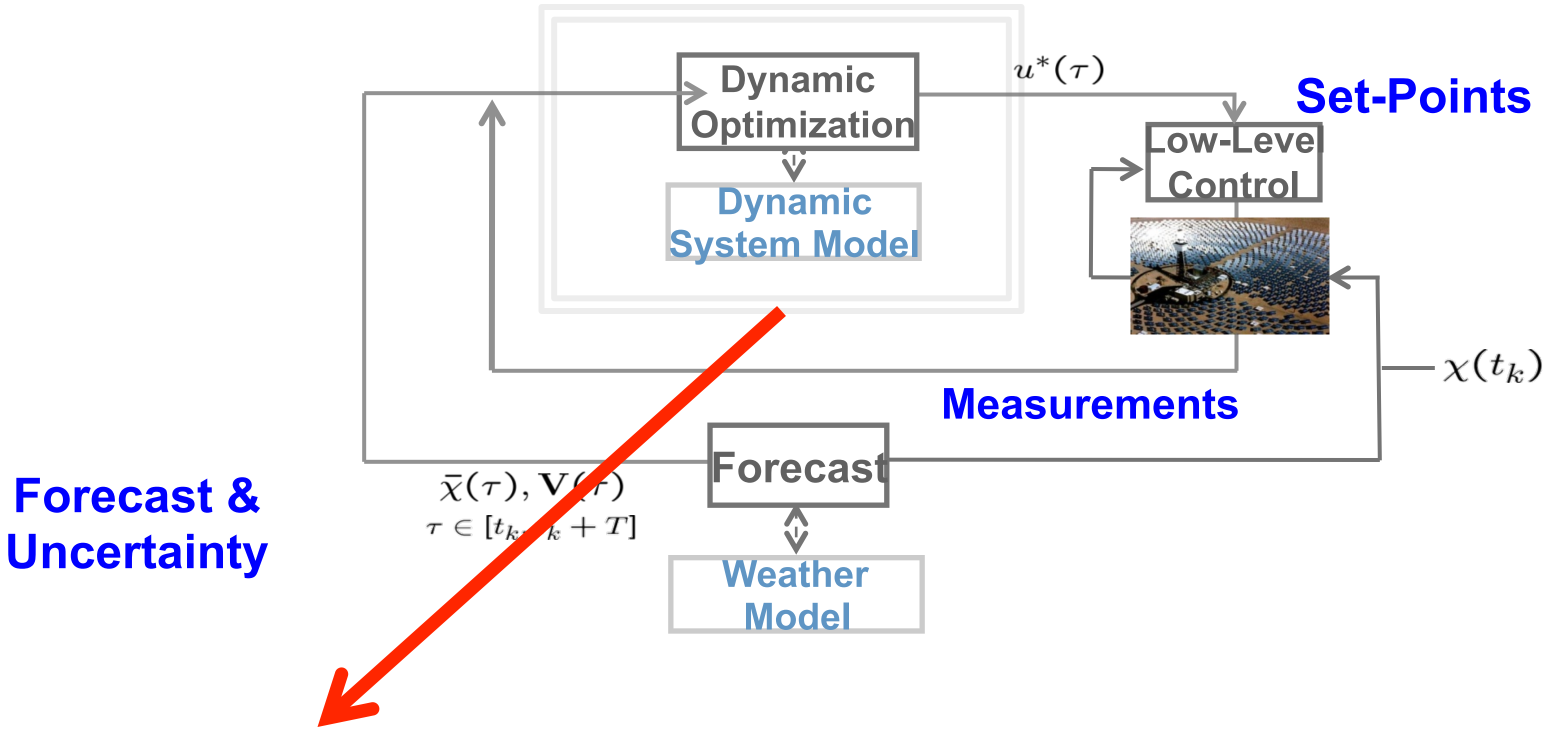
With: Victor Zavala

Big Picture



Real-Time Optimization is Pervasive in Energy : Estimation, Management, Control
Requires Extreme-Scale NLP Solvers: Model Size and Short Time Scales

Model Predictive Control (MPC)



Need for MPC

- Traditional control approximates the model based on output (mostly) ignoring its physical structure.
- High variability in forcing and nonlinearity requires a physical model-based approach.
- Far more computationally intensive bottleneck is optimization problem.

Dynamic Optimization for MPC

$$\begin{aligned} \min_{u(\tau),z(t)} \quad & \int_t^{t+T} \varphi\left(z(\tau),,u(\tau),\omega(t)\right) d\tau \\ \text{s.t.} \quad \frac{dz}{d\tau} &= \mathbf{f}(z(\tau),u(\tau),\omega(t)) \\ 0 &\geq \mathbf{h}(z(\tau),u(\tau),\omega(t)) \end{aligned}$$

Fundamental Limitations of Off-The-Shelf Optimization

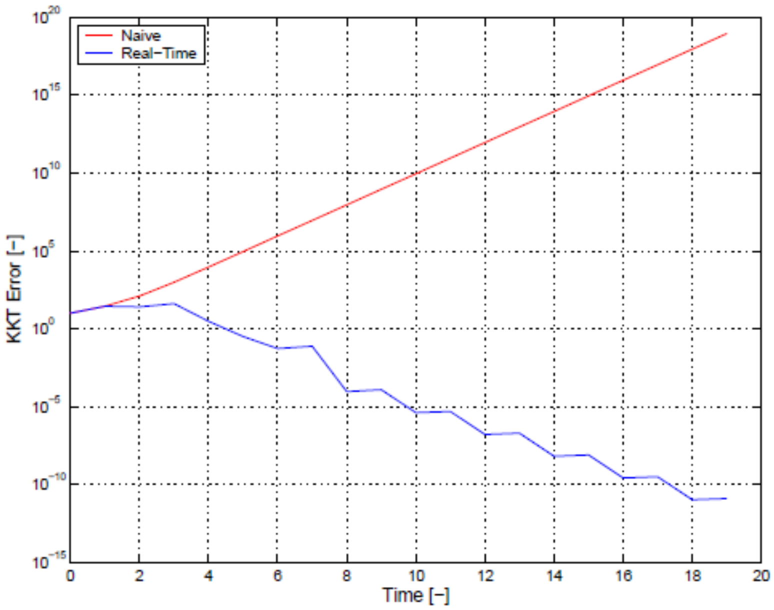
Example DO:

$$\min_{x(t)} \frac{1}{2}(x(t)-\eta(t))^2 + \frac{1}{2}x(t)^2 \cdot \eta(t)$$

Off-the-Shelf : Solve to Given Accuracy (Neglect Dynamics)

$$\epsilon^j(t) = \|\nabla_x f(x^j(t), \eta(t))\| \leq \delta_\epsilon$$

Real-Time (Z & A) : <u>One</u> SQP Iteration per step



Outline of the Talk

1. Generalized Equation / “Incomplete Optimization”

2. Exact Differentiable Penalty Approach for Accuracy and Reduced Latency

3. Numerical Case Studies

4. Conclusions and Future Work

1.Generalized Equation / “Incomplete Optimization”

MPC as Dynamic Generalized Equation (Z & A)

Context: Parametric NLP $\min_{x \in X} f(x,t), \text{ s.t. } c(x,t) = 0$

KKT system for QP

$$\begin{array}{ll} \min & \nabla_x f(x^*_{t_0}, t)^T \Delta x + \frac{1}{2} \Delta x^T \nabla_{xx} \mathcal{L}(w^*_{t_0}, t_0) \Delta x \\ \text{s.t.} & c(x^*_{t_0}, t) + \nabla_x c(x^*_{t_0}, t_0)^T \Delta x = 0 \\ & \Delta x \geq -x^*_{t_0} \end{array}$$

Time linearization of Optimality Conditions: Find $\bar{w}_t = [\bar{x}_t \; \bar{\lambda}_t]$

$$\boxed{0 \in F(w^*_{t_0}, t) + \nabla_w F(w^*_{t_0}, t_0)(w - w^*_{t_0}) + \mathcal{N}_W(w)}$$

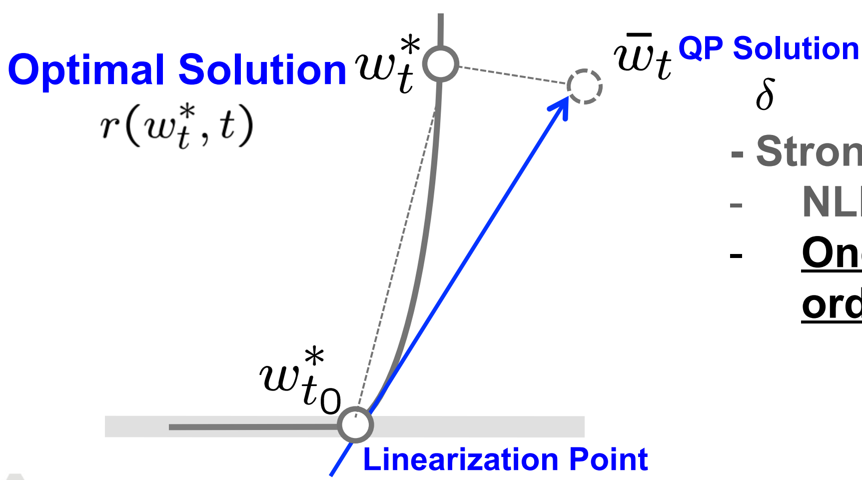
$w^*_{t_0}$

Note: Canonical Form Identical to Time-Steping for DVI

Exact Solution Satisfies:

$$\delta \in F(w^*_{t_0}, t_0) + \nabla_w F(w^*_{t_0}, t_0)(w - w^*_{t_0}) + \mathcal{N}_W(w) \qquad \delta = F(w^*_{t_0}, t_0) - F(w^*_{t_0}, t)$$

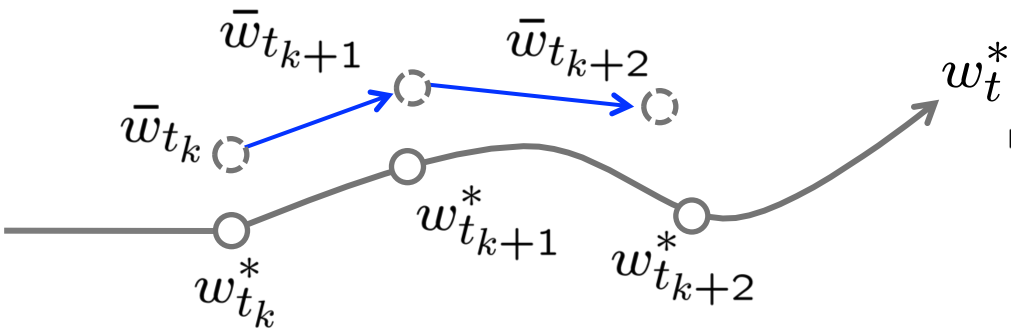
From Lipschitz Continuity of strongly regular GE: $\left\| w_t^* - \bar{w}_t \right\| \leq L \Delta t^2$



- Strong Regularity Requires SSOC and LICQ
- NLP Error is Bounded by LGE <u>Perturbation</u>
- **<u>One QP solution from exact manifold is second-order accurate</u>**

One-QP per step stabilizes

But for linearized DO I am never EXACTLY on the manifold: What then?



Solve off-manifold time-dependent QP

$$\begin{array}{ll} \min & \nabla_x f(\bar{x}_{t_k}, t_{k+1})^T \Delta x + \frac{1}{2}\Delta x^T \nabla_{xx}\mathcal{L}(\bar{w}_{t_k}, t_k)\Delta x \\ \text{s.t.} & c(\bar{x}_{t_k}, t_{k+1}) + \nabla_x c(\bar{x}_{t_k}, t_k)^T \Delta x = 0 \\ & \Delta x \geq -\bar{x}_{t_k} \end{array}$$

Theorem (elucidating an issue posed by Diehl et al.)

- **A: LGE is Strongly Regular at ALL** $w^*_{t_k}$ **e.g. NLP satisfies LICQ and SOSC everywhere**

Then: For sufficiently small Δt **, we can track the manifold stably, solving 1 QP per step**

$$\boxed{\|\bar{w}_{t_k} - w^*_{t_k}\| \leq L_\psi \delta_r \Rightarrow \|\bar{w}_{t_{k+1}} - w^*_{t_{k+1}}\| \leq L_\psi \delta_r}$$

Moreover: Stability Holds Even if QP Solved t $\mathcal{O}(\Delta t^2)$ **accuracy. Can use iterative methods.**

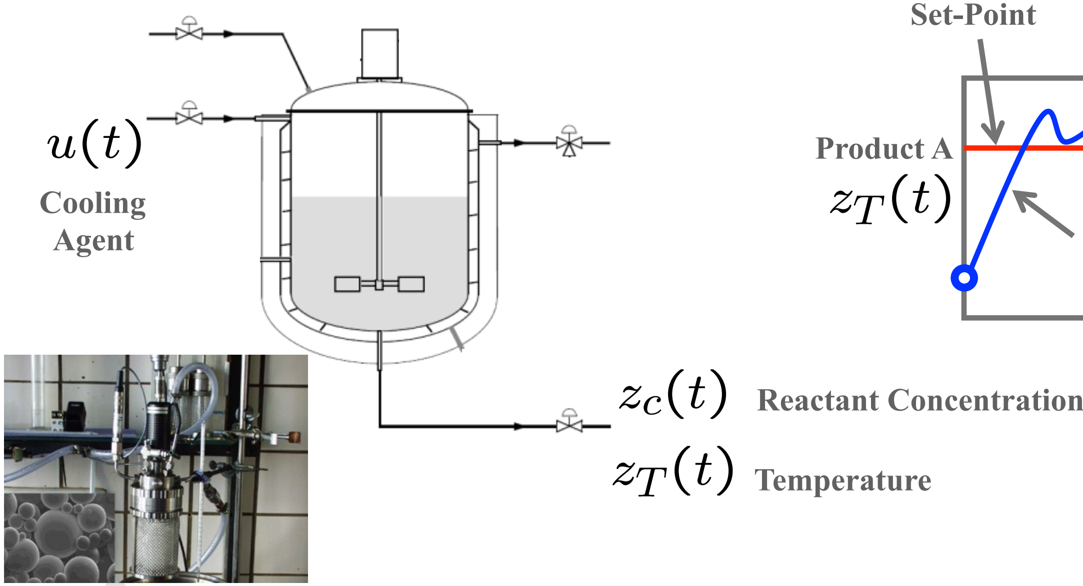
Much less effort per step and better chances for real-time performance !

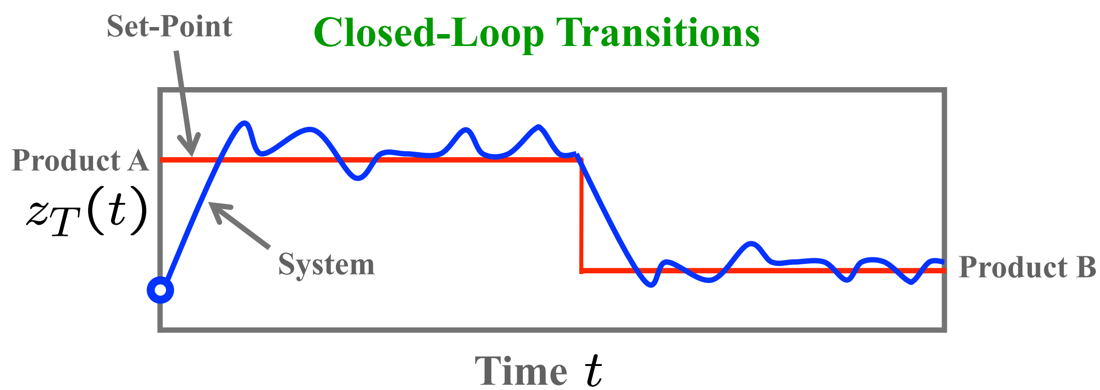
.

Need for more features of DO solvers

- One QP per step may still be too much
- Moreover I may need also good global and fast local convergence properties as well, it is not all about asymptotics!
- Sometimes one switch regimes, the optimal point moves far away, and you still want to be able to track well. – MPC algorithm must exhibit global convergence and fast local convergence (i.e. Newton)!
- Also, power grid problems can be huge (US ~ 1 – 100 Billion Variables). Need scalable solvers.

Control of Polymerization Reactor





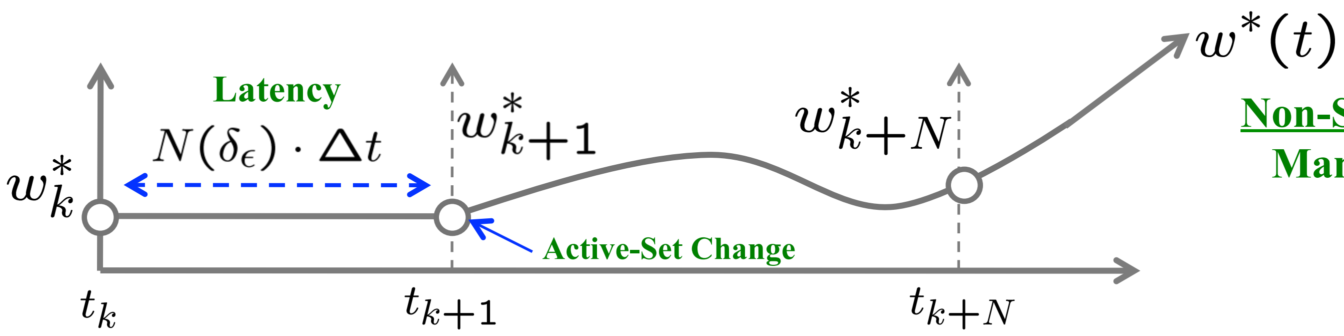
2. Exact Differentiable Penalty Approach for Accuracy and Reduced Latency

Technical Problem

$$\min_x f(x,t)$$
$$\text{s.t. } h(x,t) = 0, \quad (\lambda)$$
$$x \geq 0.$$

$$w^T = [x^T, \lambda^T]$$

Solution forms Time-Moving and Non-Smooth Manifold



Non-Smooth Manifold

- **Challenge is to Track Manifold Accurately (Classical Optimization) AND Stably (Latency Conscious: A good Step, Computer Fast)**

Technical Problem

- **Challenge is to Track Manifold Accurately AND Stably (Get Good Step with Minimum Latency)**

- **This requires NLP Solvers with the Following Features:**
  - **A) Classical Optimization Oriented :**

  1) Superlinear Convergence (Newton-Based)
  2) Scalable Step Computation (Iterative Linear Algebra)

  - **B) Latency Conscious:**

  3) Asymptotic Monotonicity of Minor Iterations (Makes Progress in O(N))
  4) Active-Set Detection and Warm-Start

- **Existing Solvers Tend to Fail at Least One Feature**
  - **Interior Point: 4, and to some extent, 2,3**
  - **Augmented Lagrangian: 1**
  - **SQP: 2**

Exact Differentiable Penalty Functions (EDPFs)

Consider Transformation using Squared Slacks

$$\begin{aligned} \min_x \;& f(x) \\ \text{s.t. } h(x) &= 0 \\ x &\geq 0 \end{aligned} \qquad\qquad \begin{aligned} \min_{x,z} \;& f(x) \\ \text{s.t. } h(x) &= 0 \\ x &= z^2 \end{aligned}$$

Equivalent To:

$$\begin{aligned} \min_z \;& f(z^2) \\ \text{s.t. } h(z^2) &= 0 \end{aligned} \qquad \mathcal{L}(z^2, \lambda) = f(z^2) + \lambda^T h(z^2)$$

$$\begin{aligned} \nabla_z \mathcal{L}(z^2, \lambda) &= 2 \cdot Z \cdot \left(\nabla f(z^2) + \nabla h(z^2)\lambda\right) \\ &= 2 \cdot X^{1/2} \nabla_x \mathcal{L}(x, \lambda) \end{aligned}$$

Apply DiPillo and Grippo's Penalty Function *DiPillo,Grippo, 1979, Bertsekas, 1982*

$$P(x, \lambda, \alpha, \beta) = \mathcal{L}(x, \lambda) + \frac{1}{2}\alpha c(x)^T c(x) + \boxed{2\beta \nabla_x \mathcal{L}(x, \lambda)^T X \nabla_x \mathcal{L}(x, \lambda)}$$

Solve NLP Indirectly Through EDPF Problem:

$$\min_{x,\lambda} \; P(x, \lambda, \alpha, \beta) \text{ s.t. } x \geq 0$$

Exact Differentiable Penalty Functions with Bound Constraints

$$\min_x \; f(x) \quad \text{s.t. } h(x) = 0, \quad x \geq 0 \quad \longleftrightarrow \quad \min_{x,\lambda} \; P(x,\lambda,\alpha,\beta) \text{ s.t. } x \geq 0$$

$$P(x,\lambda,\alpha,\beta) = \mathcal{L}(x,\lambda) + \frac{1}{2}\alpha h(x)^T h(x) + 2\beta \nabla_x \mathcal{L}(x,\lambda)^T X \nabla_x \mathcal{L}(x,\lambda)$$

Advantages

- **EDPF Differentiable Everywhere**
- **Unconstrained Problem with Box Constraints, scalable, superlinear, warm-start**
- **Makes Progress at Each Iteration (latency)**

Questions

- **Under What Conditions Do Minimizers of EDPF and NLP Coincide?**
- **How to Deal with Nonconvexity?**
  - **Detect and Exploit Negative Curvature**
- **Can We Enable Scalability AND NOT NEED THIRD DERIVATIVE?**
  - **First and Second Derivatives**
  - **Iterative Linear Algebra**

The big picture

- Combine Bertsekas bound constrained EDPF with Lin-More trust region.
- Superlinear convergence w/o Maratos from EDPF
- Matrix free from Lin-More
- Improvement in Order N from EDPF
- Warm-Start and active set detection from Lin-More
- And maybe this will help optimization proper ….

- Our contributions:
- Formalizing bound constrained EDPF properties
- Using trust-region to get rid of the third derivative while preserving both global convergence of EPF and superlinear convergence of Newton.
- Demonstrating that the approach scales well.

Derivatives and Minimizers of EDPF

$$P(x, \lambda, \alpha, \beta) = \mathcal{L}(x, \lambda) + \frac{1}{2}\alpha h(x)^T h(x) + 2\beta \nabla_x \mathcal{L}(x, \lambda)^T X \nabla_x \mathcal{L}(x, \lambda)$$

In Compact Form

$$P_{\alpha,\beta}(w) = \mathcal{L}(w) + \frac{1}{2}\nabla_w \mathcal{L}(w)^T K_{\alpha,\beta}(w) \nabla_w \mathcal{L}(w) \qquad K_{\alpha,\beta}(w) = \begin{bmatrix} 4\beta X & \\ & \alpha I_m \end{bmatrix}$$

First Derivative

$$\nabla P = \nabla \mathcal{L} + \nabla^2 \mathcal{L} K \nabla \mathcal{L} + \frac{1}{2}\Gamma \operatorname{diag}(\nabla \mathcal{L}) \nabla \mathcal{L}$$

Is KKT Point of EDPF a KKT Point of NLP?

$$\begin{matrix} \sqrt{X}\nabla_x P = 0 \\ \nabla_\lambda P = 0 \end{matrix} \longrightarrow \begin{matrix} \sqrt{X}\nabla_x \mathcal{L}(x, \lambda) = 0 \\ \nabla_\lambda \mathcal{L}(x, \lambda) = 0 \end{matrix}$$

Theorem:

Under LICQ and SC there exist $\alpha, \beta,$ such that KKT Point of EDPF is KKT point of NLP.

Proof:

$$\begin{bmatrix} \mathbb{I}_{n\times n} + 4\beta\sqrt{X}\nabla_{x,x}\mathcal{L}(w^*)\sqrt{X} + 2\beta \operatorname{diag}\left(\nabla_x \mathcal{L}(w^*)\right) & \alpha\sqrt{X}\nabla_x h(x^*)^T \\ 4\beta \nabla_x h(x^*)\sqrt{X} & \mathbb{I}_{m\times m} \end{bmatrix} \begin{bmatrix} \sqrt{X}\nabla_x \mathcal{L}(w^*) \\ h(x^*) \end{bmatrix} = \begin{bmatrix} 0_n \\ 0_m \end{bmatrix}.$$

Matrix on LHS is PD For sufficient large α and sufficiently small β .

Derivatives and Minimizers of EDPF

Second Derivative

$$\nabla^2 P \cdot u = \nabla^2 \mathcal{L} \cdot u + \nabla^2 \mathcal{L} K \nabla^2 \mathcal{L} \cdot u + \nabla^2 \mathcal{L} \mathsf{diag}(\nabla \mathcal{L}) \Gamma \cdot u + \Gamma \mathsf{diag}(\nabla \mathcal{L}) \nabla^2 \mathcal{L} \cdot u + \nabla(\nabla^2 \mathcal{L} \cdot u) K \nabla \mathcal{L}$$

Third-Order Term

High-Order Term Vanishes at KKT Point Because $K \nabla \mathcal{L} = 0.$

Is Strict Minimizer of EDPF a Strict Minimizer of NLP?

Theorem:

i) **If KKT Point satisfies SSOC for NLP then there exist** $\alpha, \beta,$ **such that it satisfies SSOC of EDPF.**

ii) **If KKT Point does not satisfy SSOC for NLP then there exist** $\alpha, \beta,$ **such that this is not a strict local minimizer of EDPF.**

Proof: **Relies on Analysis of Projected Hessian where** N **is null-space matrix.**

$$\begin{aligned} &\nu^T N^T \nabla^2 P N \nu \\ &= \begin{bmatrix} \nu_x^T N_x^T & \nu_\lambda^T \end{bmatrix} \begin{bmatrix} H & A^T \\ A & \end{bmatrix} \begin{bmatrix} N_x \nu_x \\ \nu_\lambda \end{bmatrix} \\ &\quad + \begin{bmatrix} \nu_x^T N_x^T & \nu_\lambda^T \end{bmatrix} \begin{bmatrix} H & A^T \\ A & \end{bmatrix} \begin{bmatrix} 4\beta X & 0 \\ 0 & \alpha \mathbb{I}_m \end{bmatrix} \begin{bmatrix} H & A^T \\ A & \end{bmatrix} \begin{bmatrix} N_x \nu_x \\ \nu_\lambda \end{bmatrix}. \end{aligned}$$

Derivatives and Minimizers of EDPF

A "Strong" Dennis-More Condition

Exact Hessian

$$\nabla^2 P \cdot u = \nabla^2 \mathcal{L} \cdot u + \nabla^2 \mathcal{L} K \nabla^2 \mathcal{L} \cdot u + \nabla^2 \mathcal{L} \mathrm{diag}(\nabla \mathcal{L}) \Gamma \cdot u + \Gamma \mathrm{diag}(\nabla \mathcal{L}) \nabla^2 \mathcal{L} \cdot u + \nabla(\nabla^2 \mathcal{L} \cdot u) K \nabla \mathcal{L}.$$

Approximate Hessian

$$Q \cdot u = \nabla^2 \mathcal{L} \cdot u + \nabla^2 \mathcal{L} K \nabla^2 \mathcal{L} \cdot u + \nabla^2 \mathcal{L} \mathrm{diag}(\nabla \mathcal{L}) \Gamma \cdot u + \Gamma \mathrm{diag}(\nabla \mathcal{L}) \nabla^2 \mathcal{L} \cdot u$$

Approximate Hessian is Asymptotically Convergent

$$\begin{aligned}(Q(w) - \nabla^2 P(w)) \cdot u &= \nabla(\nabla^2 \mathcal{L}(w) \cdot u) K(w) \nabla \mathcal{L}(w) \\ &= o(u) O(\|w - w^*\|), \quad \text{because} \quad K(w^*) \nabla \mathcal{L}(w^*) = 0 \\ &\overset{w \to w*}{=} 0.\end{aligned}$$

Implication:

- **We can drop third-order terms and derive quasi-Newton algorithms that retain superlinear convergence.**
- **Much easier implementation.**

Trust-Region Newton

$$\min_{x,\lambda} \; P_{\alpha,\beta}(w) \text{ s.t. } w \in \Omega$$

- Need to detect and exploit directions of negative curvature

- Use Trust-Region Newton Framework of Lin and More (TRON)

1) Determine Activity Using Cauchy Point

$$[w^c, \mathcal{A}^c] = \text{Proj}[w - \alpha^c \nabla P(w)]$$

2) Compute Search Step by Solving Trust-Region QP using Steihaug's Preconditioned Conjugate Gradient Approach (PCG)

$$\begin{aligned} \min_{\Delta w} \quad & \nabla P(w)^T \Delta w + \frac{1}{2} \Delta w^T Q(w) \Delta w \\ \text{s.t.} \quad & \Delta w_i = 0, \quad i \in \mathcal{A}^c \\ & \|\Delta w\| \leq \Delta \end{aligned}$$

3) Check Progress Over Cauchy Step and Update Trust Region Radius

- Approach Converges to Strict Local Minimizers of NLP Globally and Superlinearly
- Requires $\alpha, \beta,$ to Satisfy Conditions of Previous Theorems

Computational Scalability

Derivatives

- **EDPF Hessian Can be Assembled using Hessian and Jacobian Vector Products**

$$\nabla^2 \mathcal{L} \cdot \nu = \begin{bmatrix} H & A^T \\ A & \end{bmatrix} \begin{bmatrix} \nu_x \\ \nu_\lambda \end{bmatrix} = \begin{bmatrix} H \cdot \nu_x + A^T \cdot \nu_\lambda \\ A \cdot \nu_x \end{bmatrix}.$$

Kernel

$$Q \cdot u = \nabla^2 \mathcal{L} \cdot u + \nabla^2 \mathcal{L} K \nabla^2 \mathcal{L} \cdot u + \nabla^2 \mathcal{L} \mathrm{diag}(\nabla \mathcal{L}) \Gamma \cdot u + \Gamma \mathrm{diag}(\nabla \mathcal{L}) \nabla^2 \mathcal{L} \cdot u$$

Requires 2 Unique Kernels

PCG

$$\begin{aligned} \min_{s_d^k} \; & g^{k^T} N^k s_d^k + \frac{1}{2} s_d^{k^T} (N^k)^T Q^k N^k s_d^k \\ \text{s.t.} \; & \| D^k N_j^k s_d^k \| \leq \Delta^k. \end{aligned}$$

- **Does Not Require Assembling Reduced Hessian**
- **Requires Action of Inverse Preconditioner** $(D^k)^{-1} \cdot r$
- **Incomplete Cholesky, PARDISO, Algebraic Multigrid**
- **Inertia Detected Externally (Not by Linear Solver)**

3. Numerical Results

Numerical Examples

Algorithmic Behavior

$$\begin{aligned} \min \quad & (x_1 - 1)^2 + (x_2 - 2)^2 + (x_3 - 3)^2 + x_1 x_4 \\ \text{s.t.} \quad & x_1 x_4 + x_1 x_2 + x_3 = 4, \qquad (\lambda) \\ & x_1, x_2, x_3, x_4 \geq 0. \end{aligned}$$

					TR		Min Eig		
k	P^k	g^k_{Proj}	ρ^k	$\lVert s^k \rVert$	$\lVert \Delta^k \rVert$	$\lVert Q^k - H^k \rVert$	$\underline{\lambda}(Q^k_d)$	$\underline{\lambda}(H^k_d)$	card$(\mathcal{A}^k_P)$
0	25.150	2.0e+2							0
1	3.449	5.9e+1	+3.26	2.5e−1	261.9	2.0e+2	-2.48	-22.67	0
2	3.449	5.9e+1	-0.70	0.0e+0	523.9	5.8e+1	-2.48	-22.67	0
3	3.449	5.9e+1	-0.62	0.0e+0	131.0	5.8e+1	-2.48	-22.67	0
4	3.449	5.9e+1	-0.33	0.0e+0	32.0	5.8e+1	-2.48	-22.67	0
5	3.449	5.9e+1	-0.28	0.0e+0	8.0	5.8e+1	-2.48	-22.67	0
6	1.533	2.5e+1	+0.37	2.0e+0	2.0	5.8e+1	-2.48	-22.67	0
7	0.945	1.6e+0	+0.52	1.9e−1	2.0	2.9e+1	+0.15	-0.39	0
8	0.944	4.9e−1	+0.48	2.6e−3	4.0	1.9e+0	+0.19	+0.37	0
9	0.943	4.5e−1	+0.93	1.4e−3	4.0	4.0e−1	+0.19	+0.25	0
10	0.909	2.3e−1	+0.94	1.8e−1	8.0	3.4e−1	+0.40	+0.40	1
11	0.908	1.7e−6	+0.99	8.7e−3	16.0	3.1e−6	+0.38	+0.38	1

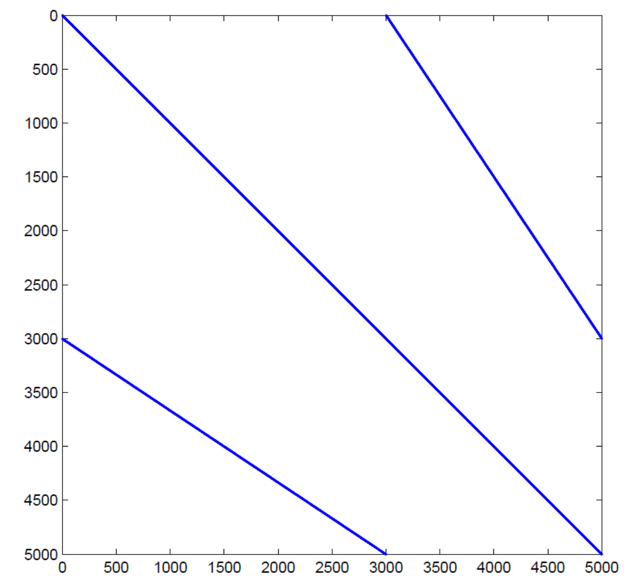
- **Trust Region Management Critical - Line Search Solvers Fail (IPOPT)**
- **High Nonlinearity at Beginning of Search (Third order term induces it)**

Numerical Examples

Optimal Control Problem

$$
\begin{aligned}
\min \quad & \int_0^T \left(\alpha_c \cdot (c(\tau) - \bar{c})^2 + \alpha_t \cdot (t(\tau) - \bar{t})^2 + \alpha_u \cdot (u(t) - \bar{u})^2\right) d\tau \\
\text{s.t.} \quad & \dot{c}(\tau) = \frac{1 - c(\tau)}{\theta} - p_k \cdot \exp\left(-\frac{p_E}{t(\tau)}\right) \cdot c(t) \\
& \dot{t}(\tau) = \frac{t_f - t(\tau)}{\theta} + p_k \cdot \exp\left(-\frac{p_E}{t(\tau)}\right) \cdot c(\tau) - p_\alpha \cdot u(\tau) \cdot (t(\tau) - t_c) \\
& c(\tau), t(\tau), u(\tau) \geq 0, \qquad \tau \in [0, T] \\
& c(0) = c(\tau_{sys}), \qquad t(0) = t(\tau_{sys}).
\end{aligned}
$$

N	n	m	n_w	nnz($\nabla^2\mathcal{L}$)	nnz(Q)	%dens($\nabla^2\mathcal{L}$)	%dens(Q)
500	1,500	1,000	2,500	10,486	26,492	2.0e−1	4.0e−1
1,000	3,000	2,000	5,000	20,996	52,972	8.4e−2	2.0e−1
5,000	15,000	10,000	25,000	104,996	264,972	1.6e−2	4.0e−2
10,000	30,000	20,000	50,000	209,996	529,972	8.3e−3	2.1e−2

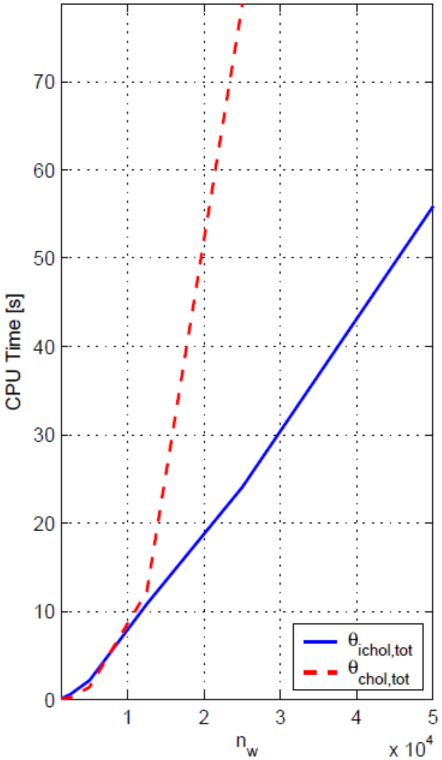


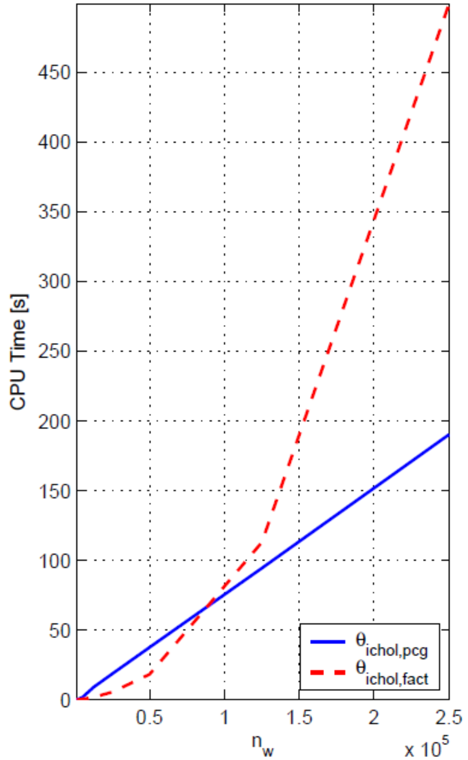
- Discretize and Scale Problem Up by Increasing Horizon N
- Sparsity of Augmented System Retained in Hessian of EDPF
- Drop Tolerance Incomplete Cholesky of 1e-4

Numerical Examples

Scalability | **Incomplete Cholesky** | **Full Cholesky**

n_w	it_{pcg}	$\theta_{ichol,pcg}$	$\theta_{ichol,fact}$	$\theta_{ichol,tot}$	$\theta_{chol,pcg}$	$\theta_{chol,fact}$	$\theta_{chol,tot}$
1,250	17	8.5e−2	3.1e−2	1.1e−1	2.7e−2	3.3e−2	6.0e−2
2,500	24	4.9e−1	1.3e−1	6.2e−1	1.1e−1	1.5e−1	2.6e−1
5,000	29	1.7e+0	4.4e−1	2.2e+0	5.7e−1	8.5e−1	1.4e+0
12,500	31	9.0e+0	1.8e+0	1.1e+1	3.8e+0	8.4e+0	1.2e+1
25,000	31	1.8e+1	5.5e+0	2.4e+1	2.5e+1	5.4e+1	7.8e+1
50,000	31	3.7e+1	1.8e+1	5.5e+1	-	-	-
125,000	31	9.4e+1	1.1e+2	2.0e+2	-	-	-
250,000	31	1.9e+2	4.9e+2	6.8e+2	-	-	-





- Scalability of Full Cholesky Not Competitive
- Incomplete Cholesky Gives High Flexibility
  Can Specify Drop Tolerance to Reduce Latency
- PCG Iterations Scale Well
- Largest Problem Has 250,000 Variables

Numerical Examples

Active-Set Identification for the 2500 dimension case

	Case 1				Case 2			
k	P^k	g^k_{Proj}	$\mathcal{A}_P(w^k)$	n^k_{PCG}	P^k	g^k_{Proj}	$\mathcal{A}_P(w^k)$	n_{PCG}
0	4.05e+3	4.52e+3	44	-	1.21e+4	2.43e+5	173	-
1	1.14e+2	4.70e+3	44	41	4.96e+2	5.76e+4	0	132
2	1.83e+1	3.72e+3	119	32	9.48e+1	1.86e+3	0	45
3	1.83e+1	1.55e+2	170	27	5.57e+0	3.27e+4	26	37
4	1.83e+1	5.59e−6	173	17	3.98e+0	1.11e+3	43	26
5	-	-	-		3.98e+0	8.50e−6	44	13

- Case 1) Has 173 variables active at solution and initialized with 44

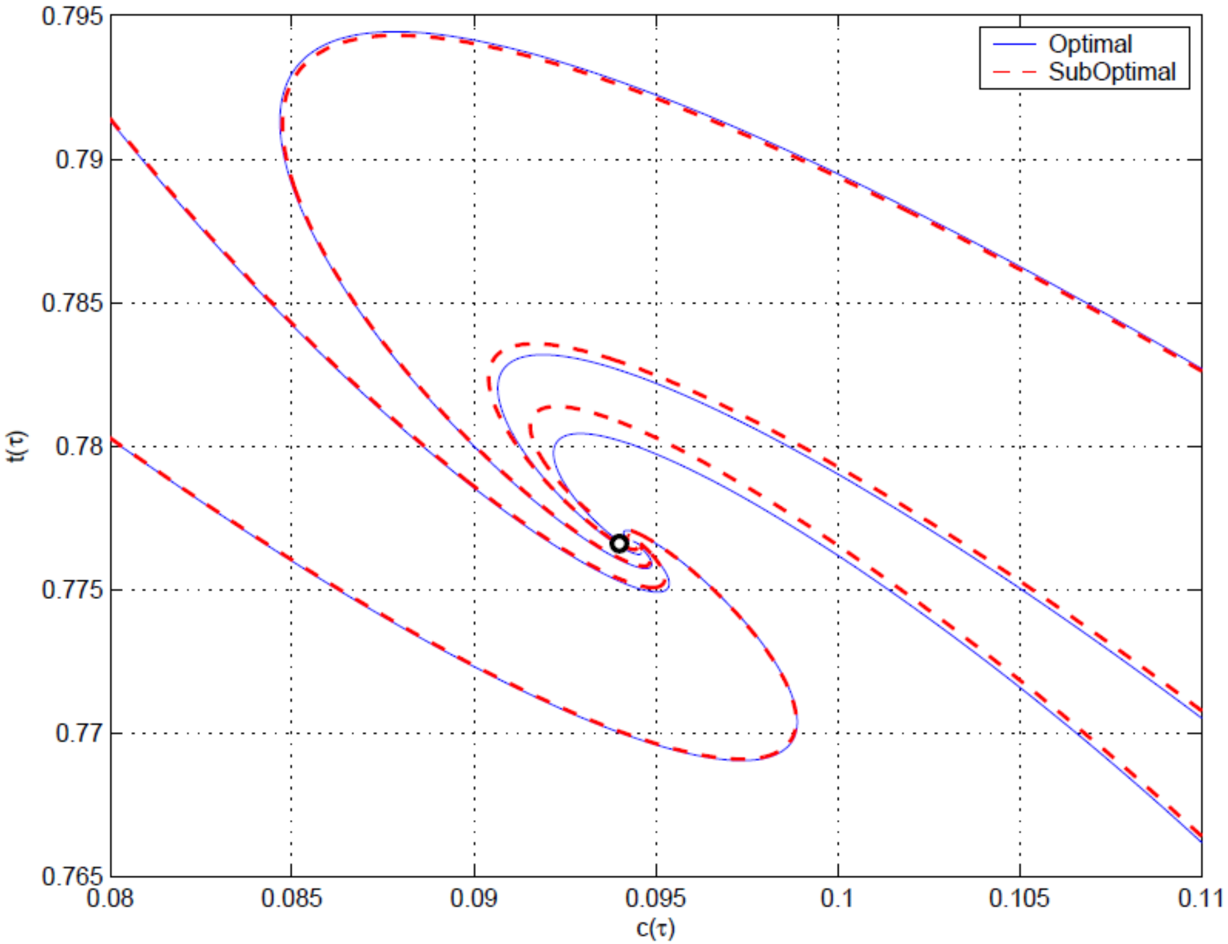
- Case 2) Has 44 variables active at solution and initialized with 173

- Cauchy Search Efficient at Detecting Activity (Allows for Large Changes Between Iterates)

- Number of PCG Iterations Do Not Degrade as Solution Approached (Compare with IP)

Numerical Examples

Early Termination on problem with N=100



- Run MPC Problem Terminating After 2 Major Iterations and 20 PCG iterations
- Reduced Latency by A Factor of 4 (Four)
- Convergence to Equilibrium Point (Warm-Starting Effective)

4. Conclusions and Future Work

- We derived NLP algorithms that enable:
  1) Superlinear Convergence (Newton-Based)
  2) Scalable Step Computation (Enable Iterative Linear Algebra)
  3) Asymptotic Monotonicity of Minor Iterations (Makes Progress)
  4) Active-Set Detection and Warm-Start

- Critical in "Fast" Real-Time Environments

- Proposed Approach : EDPF + Trust-Region Newton + PCG
  1) Newton-Based in Primal/Dual Space with Convergent Approximate Hessian
  2) Steihaug's PCG to Detect and Exploit Negative Curvatur
  3) PCG Improvement on EDPF Function
  4) Cauchy

- ToDo:
  - More Robust Implementation (Scaling, Trust-Region Update Rules, Ill-Conditioning)
  - Alternative Penalty Functions Requiring Only One Parameter
  - Preconditioning
  - Exploiting Special Structures
  - Comparison with Other NLP Solvers